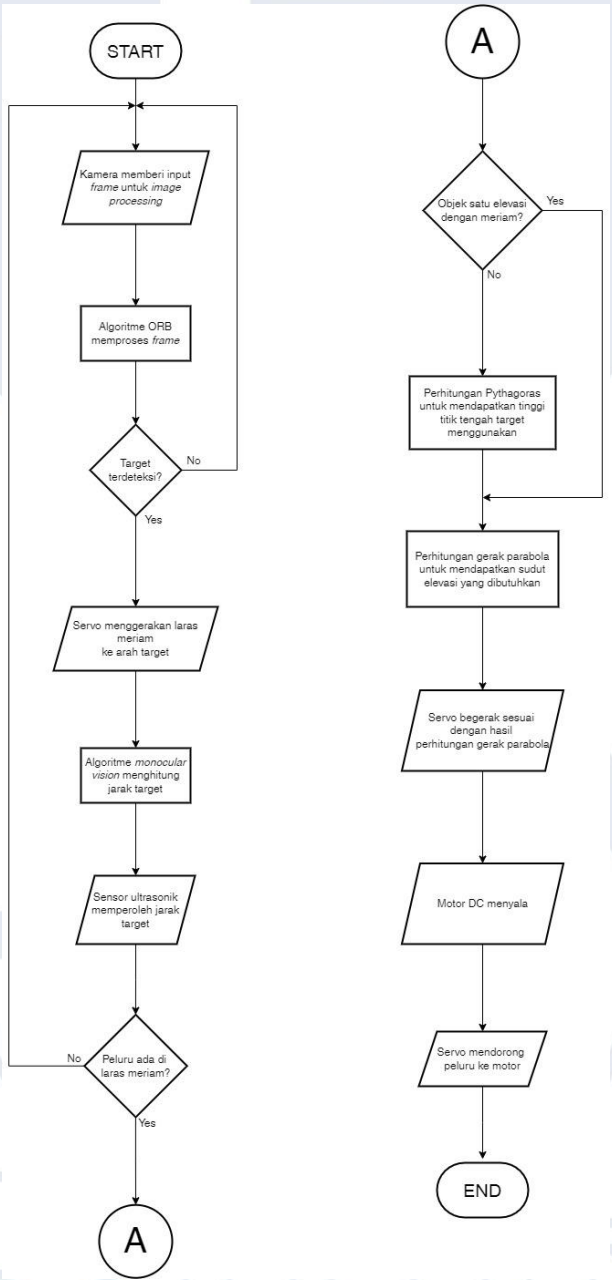
BAB III
PERANCANGAN DAN IMPLEMENTASI SISTEM

3.1 Tinjauan Desain Sistem

Sistem terdiri dari empat subsistem. Subsistem tersebut adalah subsistem pengolah gambar digital, subsistem akuisisi jarak sensor ultrasonik, subsistem kontrol meriam, dan subsistem pelontar. Keempat subsistem bekerja sama untuk mencapai tujuan menembak target secara otomatis.

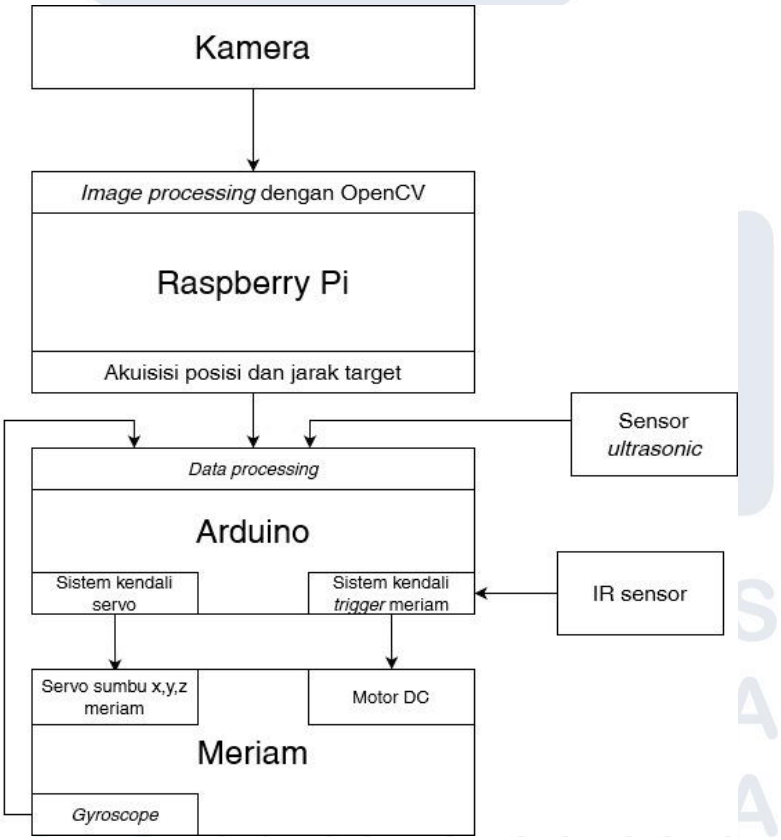


Gambar 3.1 Flowchart Sistem

Di *flowchart* pada gambar 3.1, alur kerja sistem diawali dengan kamera mengambil gambar. Gambar ini akan digunakan sebagai input untuk algoritme ORB, dan *monocular vision*. Jika algoritme ORB mendeteksi target, koordinat target dijadikan input untuk gerakan *servo*. Setelah *servo* menggerakan laras meriam ke arah target, sensor ultrasonik dan algoritme *monocular vision* menghitung jarak antara meriam dan target. Jika peluru ada di meriam, perhitungan gerak parabola akan dimulai. Jika elevasi target lebih tinggi dibandingkan sistem, dilakukan perhitungan untuk mendapatkan tinggi target dari lantai menggunakan jarak yang diperoleh *monocular vision*, dan sensor ultrasonik. Setelah tinggi target diperoleh, sudut elevasi yang dibutuhkan meriam untuk menembak target dihitung, lalu *servo* sumbu x digerakan sesuai hasil perhitungan. Motor DC kemudian dinyalakan hingga motor berputar dengan kecepatan maksimalnya, lalu peluru didorong ke motor DC.

3.1.1 Desain Sistem Keseluruhan

Desain sistem secara keseluruhan dapat dilihat di gambar 3.2



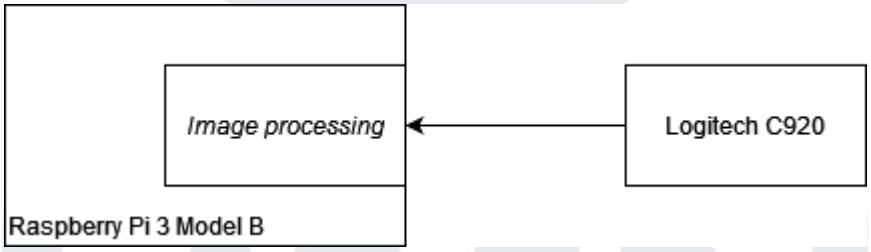
Gambar 3.2 Desain Sistem Secara Keseluruhan

Gambar 3.2 menggambarkan desain sistem secara keseluruhan. Sistem menggunakan Arduino Mega, dan Raspberry Pi 3 Model B yang saling berhubungan secara *serial*. Raspberry Pi terhubung dengan kamera dan berfungsi untuk menjalankan algoritme ORB, dan *monocular vision*. Hasil algoritme diteruskan ke Arduino sebagai input untuk sistem kendali *servo*. Selain dari Raspberry Pi, Arduino terhubung dengan sensor ultrasonik untuk mendapatkan jarak, *infrared* untuk mengecek isi meriam, dan *gyroscope* untuk membaca posisi meriam.

3.1.2 Desain Subsistem

Subsistem terdiri dari subsistem pengolah gambar digital, subsistem akuisisi jarak sensor ultrasonik, subsistem kontrol meriam, dan subsistem pelontar. Tiap subsistem memiliki komponennya masing-masing, dan memiliki fungsi yang berbeda-beda.

3.1.2.1 Desain Subsistem Pengolah Gambar Digital



Gambar 3.3 DFD Level 2 Subsistem Pengolah Gambar Digital

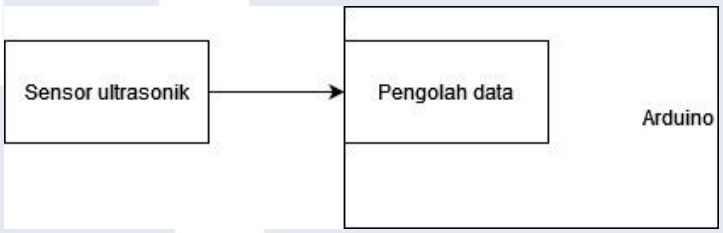
Tabel 3.1 Penjelasan DFD Level 2 Pengolah Gambar Digital

Parameter	Keterangan
Input	• Data gambar dari kamera
Output	• Posisi target • Jarak antara meriam dengan target dalam satuan meter
Fungsi	• Gambar dari kamera diproses dengan metode ORB untuk memperoleh posisinya. • Gambar dari kamera diproses menggunakan algoritme *monocular vision* untuk memperoleh jarak antara meriam dengan kamera.

Gambar 3.3 menggambarkan hubungan komponen pada subsistem pengolah gambar digital. Subsistem ini berperan untuk memperoleh posisi

dan jarak dari meriam ke target. *Input* gambar diperoleh melalui kamera logitech C920, dan untuk melakukan *image processing*, digunakan raspberry pi 3 model b. Proses gambar yang dilakukan adalah penguncian target menggunakan algortime ORB, dan akuisisi jarak menggunakan *monocular vision*.

3.1.2.2 Desain Subsistem Akuisisi Jarak Sensor Ultrasonik



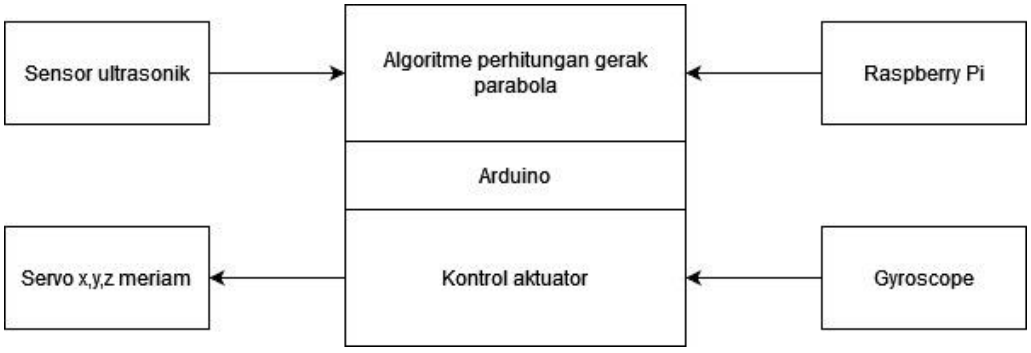
Gambar 3.4 DFD Level 2 Subsistem Akuisisi Jarak Sensor Ultrasonik

Tabel 3.2 Penjelasan DFD Level 2 Subsistem Akuisisi Jarak Sensor Ultrasonik

Parameter	Keterangan
Input	• Durasi gelombang ultrasonik
Output	• Jarak antara meriam dengan target dalam satuan meter
Fungsi	• Menghitung jarak antara meriam dan target

Gambar 3.4 menggambarkan hubungan komponen pada subsistem akuisisi jarak sensor ultrasonik. Subsistem ini berperan untuk mendapatkan jarak dari sensor ultrasonik. Subsistem ini menggunakan JSN-SR04T. yang terhubung ke Arduino Mega dengan metode *serial communication*. Karena pengukuran JSN-SR04T berupa durasi gelombang ultrasonik di udara, dilakuakn perhitungan terlebih dahulu untuk mengubah waktu gelombang bergerak ke jarak antara meriam dan target.

3.1.2.3 Desain Subsistem Kontrol Meriam



Gambar 3.5 DFD Level 2 Subsistem Kontrol Meriam

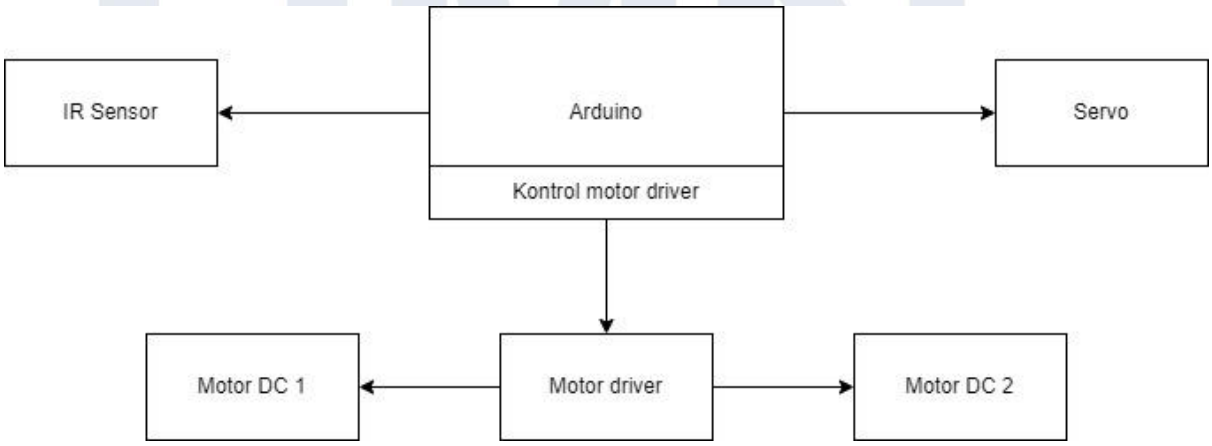
Tabel 3.3 Penjelasan DFD Level 2 Subsistem Pengolah Gambar Digital

Parameter	Keterangan
Input	• Jarak target dari meriam yang diperoleh *monocular vision* dari Raspberry Pi • Jarak target dari meriam yang diproses dari sensor ultrasonik • Posisi *gyroscope*
Output	• Gerakan *servo* x,y,z meriam
Fungsi	• Menghitung pengaturan-pengaturan yang dibutuhkan meriam agar peluru dapat mengenai target • Menggerakan *servo* x,y,z meriam berdasarkan hasil perhitungan algoritme

Gambar 3.5 menggambarkan hubungan komponen pada subsistem kontrol meriam. Subsistem berperan untuk menggerakan meriam. Subsistem ini terdiri dari tiga MG966R sebagai penggerak meriam pada sumbu x,y,z dan *gyroscope* untuk mendapatkan posisi meriam. Komponen di subsistem ini dikendalikan oleh Arduino Mega dengan input dari kedua subsistem sebelumnya.

Dengan jarak yang diperoleh dari kedua subsistem sebelumnya, diperoleh jarak antara meriam dengan target untuk dimasukan ke perhitungan gerak parabola. Dari hasil perhitungan ini, akan diperoleh sudut yang dibutuhkan meriam untuk mengenai target.

3.1.2.4 Desain Subsistem Pelontar



Gambar 3.6 DFD Level 2 Subsistem Pelontar

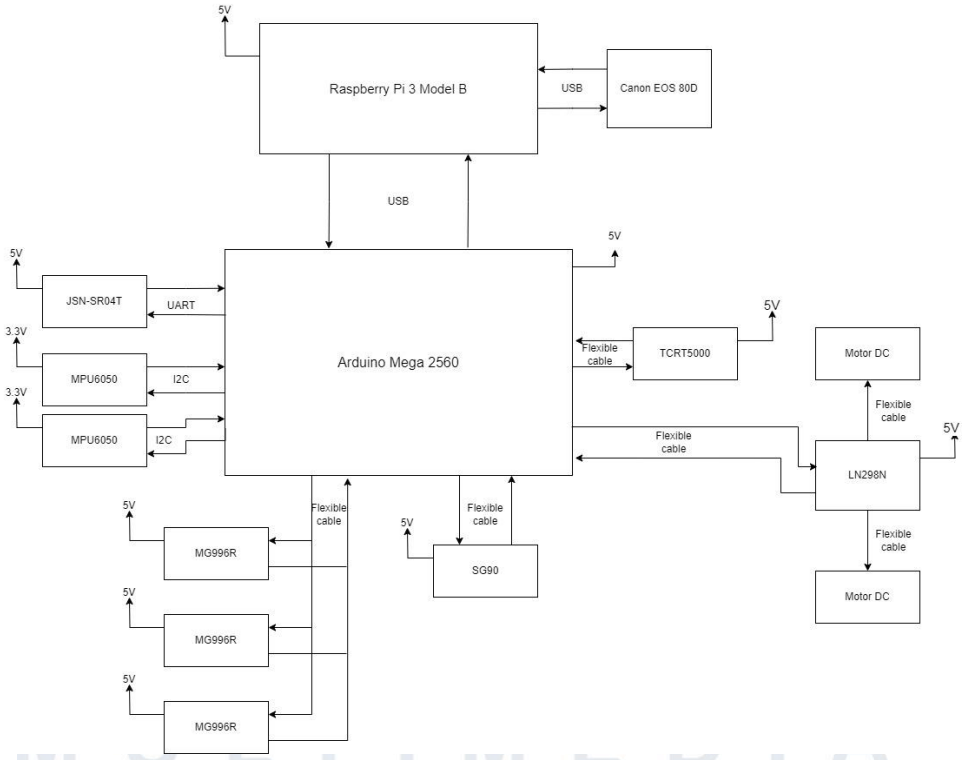
Tabel 3.4 Penjelasan DFD Level 2 Subsistem

Parameter	Keterangan
Input	• Sinyal dari IR sensor
Output	• Tegangan ke motor DC
Fungsi	• Menyalakan motor DC untuk melontarkan peluru • Mengecek adanya peluru atau tidak

Gambar 3.6 menggambarkan hubungan komponen pada subsistem pelontar. Subsistem ini berperan untuk melontarkan peluru. Subsistem ini terdiri dari dua motor DC untuk melontarkan pelurunya, *motor driver* LN298N untuk mengendalikan tegangan yang mengarah ke motor DC, sensor TCRT5000 untuk mendeteksi peluru di meriam, dan Arduino sebagai mikrokontroler untuk mengendalikan *motor driver*.

Ketika meriam sudah berada pada posisi yang dibutuhkan, motor DC akan berputar dan melontarkan peluru yang ada pada meriam. Pengendalian motor DC dilakukan melalui *motor driver* LN298N.

3.1.3 Diagram Sistem



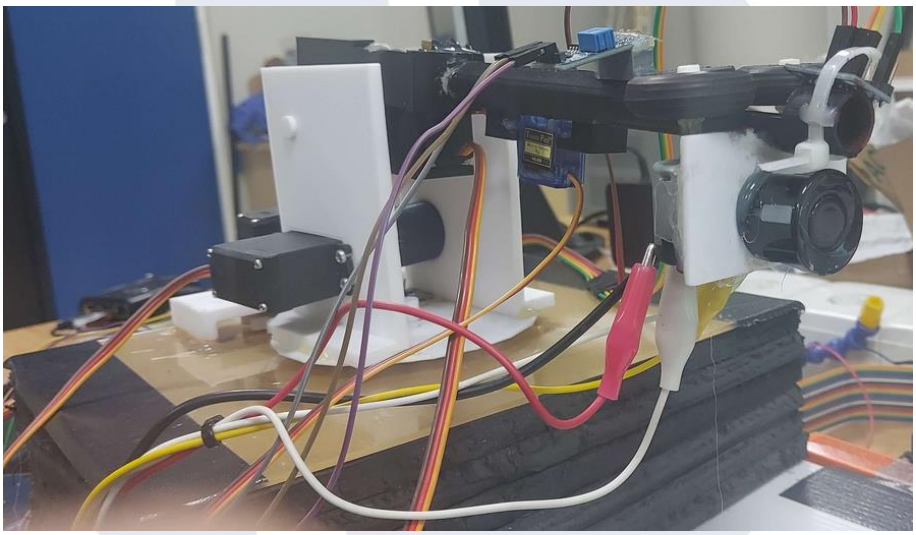
Gambar 3.7 Wiring Diagram Utama

Sistem memiliki *wiring* seperti pada gambar 3.7. MPU6050 berkomunikasi dengan Arduino Mega melalui I2C, JSN-SR04T melalui

UART, dan Raspberry Pi 3 melalui *serial* USB. TCRT5000, L298N, SG90, dan MG996R terhubung ke pin *digital* Arduino Mega.

3.2 Implementasi Sistem

3.2.1 Hasil Implementasi



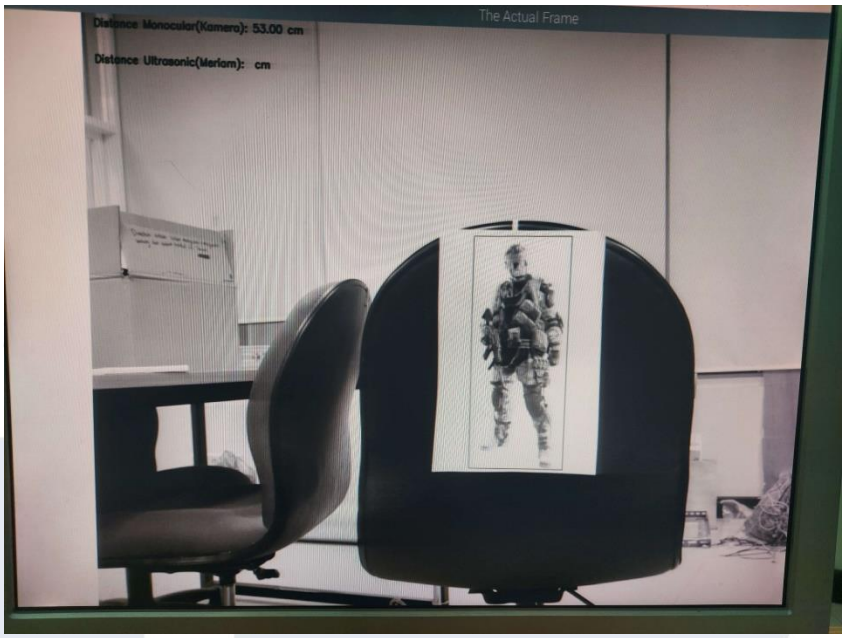
Gambar 3.8 *Prototype* Sistem

Hasil implementasi sistem dapat dilihat pada gambar 3.8. Badan produk dibuat menggunakan 3D *printer*, dan sebuah papan akrilik digunakan sebagai alas. Produk diletakan di atas tumpukan busa ati, dan *pvc board* agar laras meriam tidak kena kamera, dan agar kamera memiliki *field of view* yang lebih baik. *Gear* dipasang *ball bearing* untuk mengurangi gesekan ketika *servo* bergerak. Peluru yang digunakan juga dibuat menggunakan 3D *printer*. Peluru memiliki panjang 5 cm, diameter 1.1 cm, dan didesain dengan ujung yang memiliki luas seminimal mungkin untuk mengurangi resistansi udara seperti pada gambar 3.9.

Gambar 3.9 Peluru

3.2.1.1 Hasil Implementasi Subsistem Pengolah Gambar Digital

Subsistem pengolah gambar digital diimplementasikan dengan menjalankan program Python di Raspberry Pi 3 untuk memproses gambar yang diambil dari kamera. Program Python menggunakan *library OpenCV2* untuk menjalankan algoritme ORB, dan *monocular vision*. Program diatur untuk memproses input dengan besar *frame* 1280x720. Algoritme ORB juga diberikan pengaturan awal berupa *keypoint* maksimal sebanyak 2000, *scaling* 1.2, dan minimal *keypoint* cocok sebanyak 10. Setelah jumlah minimal *keypoint* yang cocok tercapai, sebuah kotak akan digambar di *frame* untuk menunjukan target seperti pada gambar 3.10.



Gambar 3.10 Pengujian Deteksi ORB

Dari hasil deteksi ORB, jarak objek diperoleh menggunakan algoritme *monocular vision*. Jarak diperoleh dengan rumus:

$$jarak(cm) = \frac{real\ width\ \times focal\ length}{pixel\ width} \quad (3.1)$$

dimana *real width* adalah lebar asli objek, *focal length* sebagai spesifikasi fokus kamera, dan *pixel width* berupa lebar objek dalam *pixel*. *Focal length*, dan *real width* harus diinput secara manual ke program sesuai dengan spesifikasi kamera, dan lebar asli target. Untuk mendapatkan *pixel width* digunakan fungsi "np.linalg.norm(tl[0] – tr[0])" dari *library* numpy. Dengan fungsi ini, dapat dihitung selisih vektor *top left* dan *top right* objek yang terdeteksi untuk diperoleh lebar pixelnya. Hasil perhitungan kemudian dimasukan ke "distance_list.append(distance_cm)". Jarak yang ada di daftar "distance_list" kemudian dicari rata-ratanya dengan menghitung jumlah semua isi daftar dibagi panjang daftar.

3.2.1.2 Hasil Implementasi Subsistem Akuisisi Jarak Sensor Ultrasonik

Subsistem akuisisi jarak sensor ultrasonik diimplementasikan dengan menghubungkan sensor SR-04T ke Arduino Mega pada pin RX/TX. Kode untuk menjalankan sensor ultrasonik diawali dengan *setup* pin, variabel,

dan fungsi *input/output* untuk pin yang digunakan. Gambar 3.11 menunjukan kode yang digunakan untuk memperoleh jarak. Pertama, sensor ultrasonik mengeluarkan gelombang ultrasonik. Kemudian, gelombang yang diterima dijadikan *output* ke Arduino Mega dalam bentuk *pulse*. Panjang *pulse* adalah waktu gelombang dikeluarkan, dan diterima oleh sensor dalam *microsecond*.

```
digitalWrite(trigPin, LOW);
delayMicroseconds(5);
// Trigger the sensor by setting the trigPin high for 10 microseconds:
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// Read the echoPin. pulseIn() returns the duration (length of the pulse) in microseconds:
duration = pulseIn(echoPin, HIGH);
```

Gambar 3.11 Kode yang Mengirim dan Menerima Gelombang

Dari durasi yang diterima, dimasukan ke perhitungan untuk mengubah waktu gerak gelombang, ke jarak antara sensor dan target. Rumus yang digunakan adalah:

$$Jarak = \frac{durasi(\mu s) \times 0.03523}{2} \quad (3.2)$$

Durasi didapatkan dari sensor ultrasonik, kemudian dikalikan dengan kecepatan rambat gelombang suara di udara. Karena gelombang bergerak dari sensor, ke target, lalu kembali lagi ke sensor, hasil perkalian dibagi dua. Jarak yang didapatkan kemudian diberikan ke Raspberry Pi secara *serial* untuk ditampilkan di layar *program*.

3.2.1.3 Hasil Implementasi Subsistem Kontrol Meriam

Subsistem kontrol meriam diimplementasikan dengan menghubungkan tiga *servo* MG996R, dan dua MPU6050 ke Arduino Mega. MG996R dihubungkan ke pin *digital*, sedangkan MPU6050 dihubungkan ke pin SCL/SDA. MPU6050 diletakan di badan meriam, dan di ujung laras meriam.

Subsistem kontrol meriam memiliki PID untuk menjaga kestabilan posisi meriam. Fungsi alih yang digunakan untuk PID adalah:

$$G(s) = \frac{\theta(s)}{E(s)} = \frac{K_{tn}}{s[(R_a + sL_a)(Js + f_o) + K_T K_B]} \quad (3.3)$$

Dimana K_{tn} adalah konstanta torsi motor, R_a adalah resistansi jangkar, L_a berupa induktansi kumparan jangkar (H), J adalah momen inersia motor & beban, f_o adalah koefisien gesekan viskos poros motor dan beban (Nm/rad/sec), dan K_B adalah konstanta emf lawan.

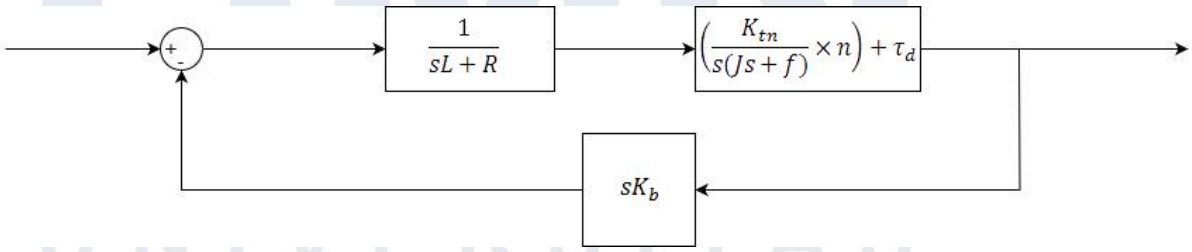
Tiap servo memiliki beban yang berbeda, sehingga nilai torsi *disturbance* pada tiap servo akan berbeda. Servo x menanggung massa 80 gram dengan panjang 17.1cm. Servo y menanggung massa 55 gram dengan panjang 6.6 cm. Dan servo z menanggung beban servo x yang memiliki massa 55 gram dengan panjang 2 cm, servo y yang memiliki massa 55 gram dengan panjang 6 cm, dan meriam dengan massa 80 gram dengan panjang 17.1 cm. Dengan massa dan panjang tiap servo, dapat diperoleh torsi *disturbance* tiap servo:

$$\tau_x = 0.055 \times 9.8 \times \frac{0.066}{2} = 0.035574 \quad (3.4)$$

$$\tau_y = 0.083 \times 9.8 \times \frac{0.171}{2} = 0.0671104 \quad (3.5)$$

$$\tau_z = 0.0671104 + 0.035574 + (0.055 \times 9.8 \times \frac{0.02}{2}) = 0.113464 \quad (3.6)$$

Dengan torsi *disturbance* yang didapat, dapat dicari fungsi alih menggunakan diagram blok seperti di gambar 3.12



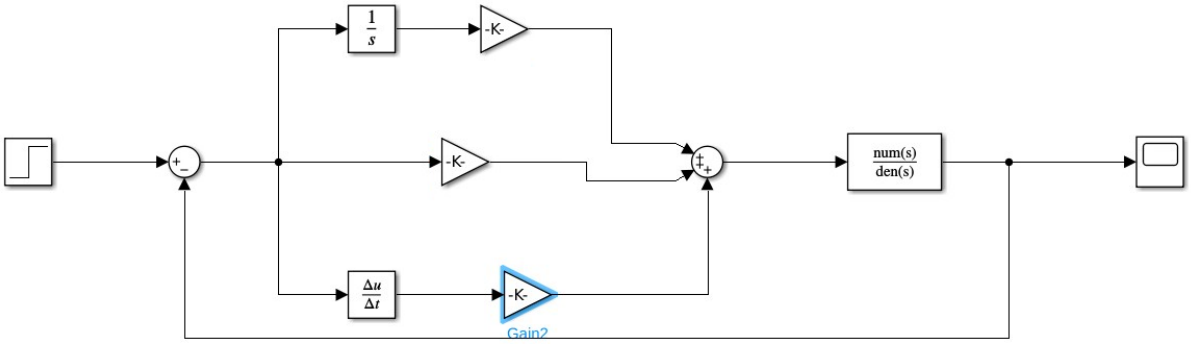
Gambar 3.12 Diagram Blok Fungsi Alih

Fungsi alih yang diperoleh adalah:

$$TF_x = \frac{0.0000001423s^3 + 0.00003577s + 0.117}{0.000002508s^3 + 0.0001627s^2 + 0.00484s} \quad (3.7)$$

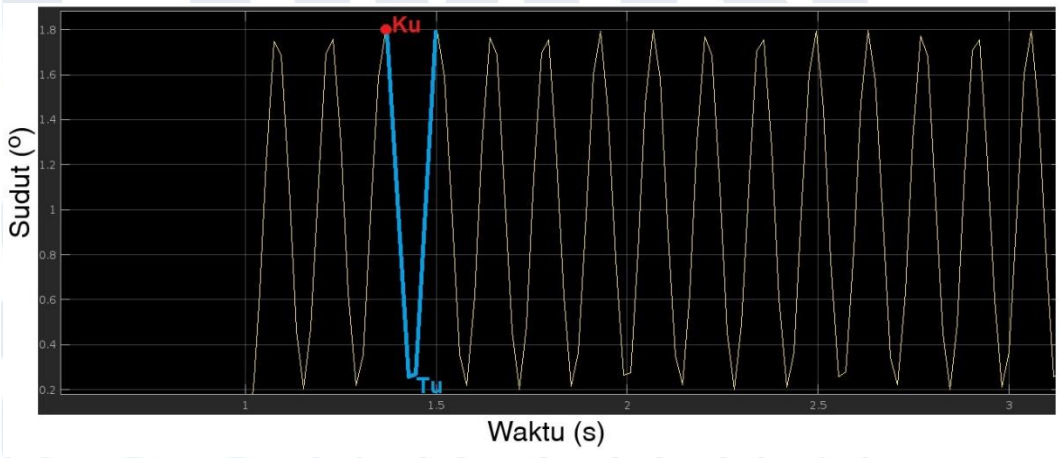
$$TF_y = \frac{0.000002684s^2 + 0.00006711s + 0.117}{0.000002534s^3 + 0.0001633s^2 + 0.00484s} \quad (3.8)$$

$$TF_z = \frac{0.000004539s^2 + 0.00001135s + 0.117}{0.000002571s^3 + 0.0001643s^2 + 0.00484s} \quad (3.9)$$



Gambar 3.13 Diagram Blok PID

Dengan fungsi alih yang diperoleh, diagram blok PID (*Proportional-integral-derivative*) dibuat menggunakan aplikasi *Simulink* seperti pada gambar 3.13. Untuk menggunakan PID, tiap komponen PID harus dikonfigurasi untuk mendapatkan respon yang diinginkan. Metode konfigurasi yang digunakan untuk mendapatkan nilai komponen yang digunakan adalah Ziegler Nichols 2. Pada metode Ziegler Nichols 2, nilai kp, ki, dan kd diatur menjadi 0. Kemudian, nilai kp dinaikkan hingga *output* sistem menghasilkan osilasi seperti pada gambar 3.14. Nilai kp ini kemudian akan digunakan sebagai Ku(*ultimate gain*). Dari grafik osilasi, dicari panjang satu gelombang untuk mendapatkan nilai Tu(*ultimate period*).



Gambar 3.14 Osilasi PID

Dengan mengatur Kp tiap PID hingga osilasi, lalu dicari *ultimate gain*, dan *ultimate period* dari *step response*, diperoleh nilai berikut:

$$Ku_x = 2.807$$
$$Tu_x = 0.1378$$
$$Ku_y = 2.907$$
$$Tu_y = 0.137$$
$$Ku_z = 3.0751$$
$$Tu_z = 0.135$$

Dengan nilai Ku dan Tu yang diperoleh, dapat dicari nilai Kp, Ki, dan Kd untuk tiap servo. Perhitungan nilai komponen PID dapat dilakukan menggunakan tabel berikut:

Tabel 3.5 Perhitungan PID Metode Ziegler-Nichols 2

Jenis pengendali	Kp	Ti	Td	Ki	Kd
P	0.5Ku	-	-	-	-
PI	0.45Ku	Tu/1.2	-	0.54Ku/Tu	-
PD	0.8Ku	-	Tu/8	-	KuTu/10
PID	0.6Ku	Tu/2	Tu/8	1.2Ku/Tu	3KuTu/40
Pessen Integral Rule	7Ku/10	2Tu/5	3Tu/20	1.75Ku/Tu	21KuTu/200
Some Overshoot	Ku/3	Tu/2	Tu/3	0.667Ku/Tu	KuTu/9
No Overshoot	Ku/5	Tu/2	Tu/3	(2/5)Ku/Tu	KuTu/15

Dengan tabel 3.5, diperoleh konfigurasi PID tiap servo sebagai berikut:

Tabel 3.6 PID Servo X

Jenis pengendali	Kp	Ti	Td	Ki	Kd
P	1.4035	-	-	-	-
PI	1.26315	0.114833	-	10.99985	-
PD	2.2456	-	0.017225	-	0.03868
Classic PID	1.6842	0.0689	0.017225	24.44412	0.02901
Pessen Integral rule	1.9649	0.05512	0.02067	35.64768	0.040614
Some overshoot	0.935667	0.0689	0.045933	13.58686	0.042978
No Overshoot	0.5614	0.0689	0.045933	8.148041	0.025787

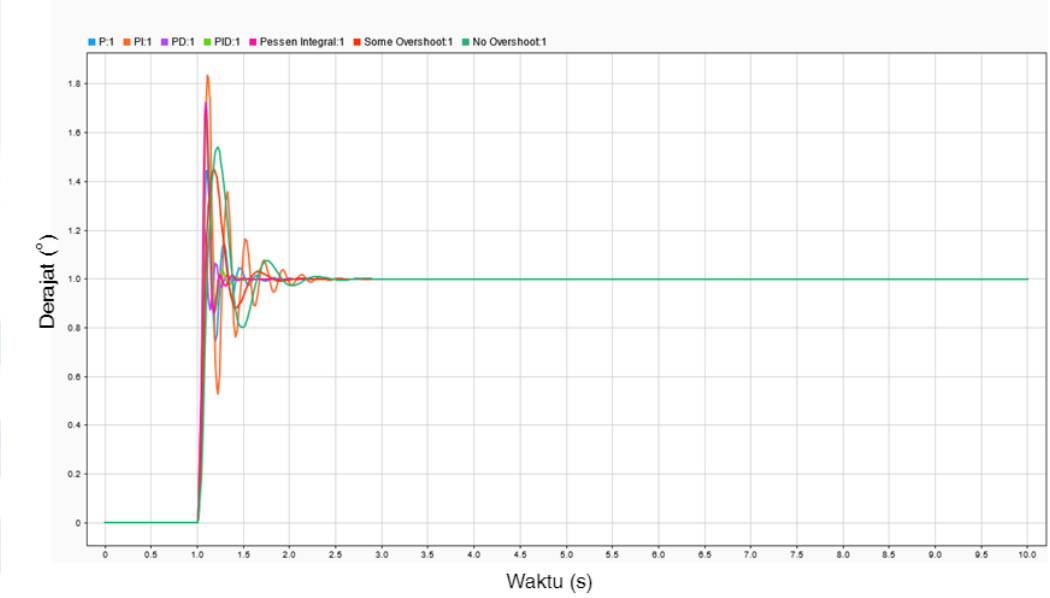
Tabel 3.7 PID Servo Y

Jenis pengendali	Kp	Ti	Td	Ki	Kd
P	1.4535	-	-	-	-
PI	1.30815	0.114167	-	11.45825	-
PD	2.3256	-	0.017125	-	0.039826
Classic PID	1.7442	0.0685	0.017125	25.46277	0.029869
Pessen Integral rule	2.0349	0.0548	0.02055	37.13321	0.041817
Some overshoot	0.969	0.0685	0.045667	14.15306	0.044251
No Overshoot	0.5814	0.0685	0.045667	8.487591	0.026551

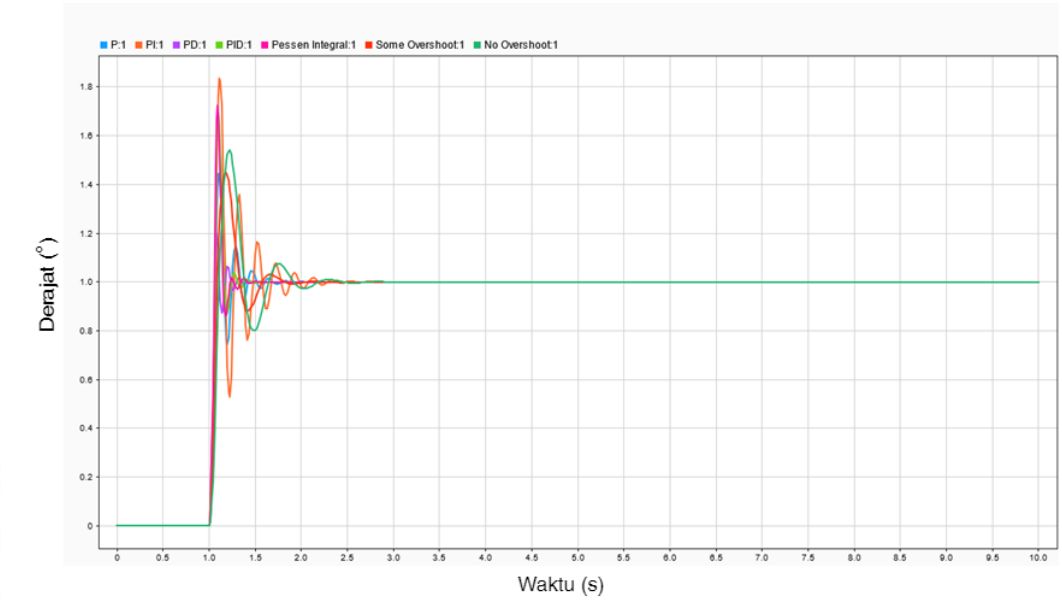
Tabel 3.8 PID Servo Z

Jenis pengendali	Kp	Ti	Td	Ki	Kd
P	1.53755	-	-	-	-
PI	1.383795	0.1125	-	12.3004	-
PD	2.46008	-	0.016875	-	0.041514
Classic PID	1.84506	0.0675	0.016875	27.33422	0.031135
Pessen Integral rule	2.15257	0.054	0.02025	39.86241	0.04359
Some overshoot	1.025033	0.0675	0.045	15.19327	0.046127
No Overshoot	0.61502	0.0675	0.045	9.111407	0.027676

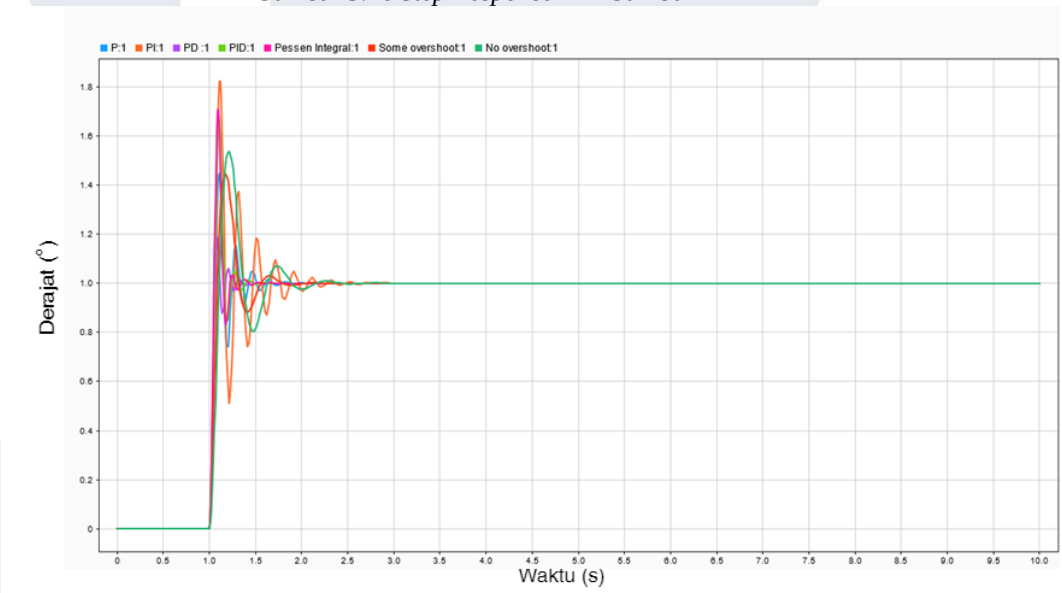
Dari tabel 3.6 hingga tabel 3.8, didapatkan tujuh jenis pengendali. Tiap pengendali memiliki *response* yang berbeda. Hasil *step response* tiap jenis pengendali dapat dilihat pada gambar 3.15 – gambar 3.17.



Gambar 3.15 *Step Response* PID Sumbu X

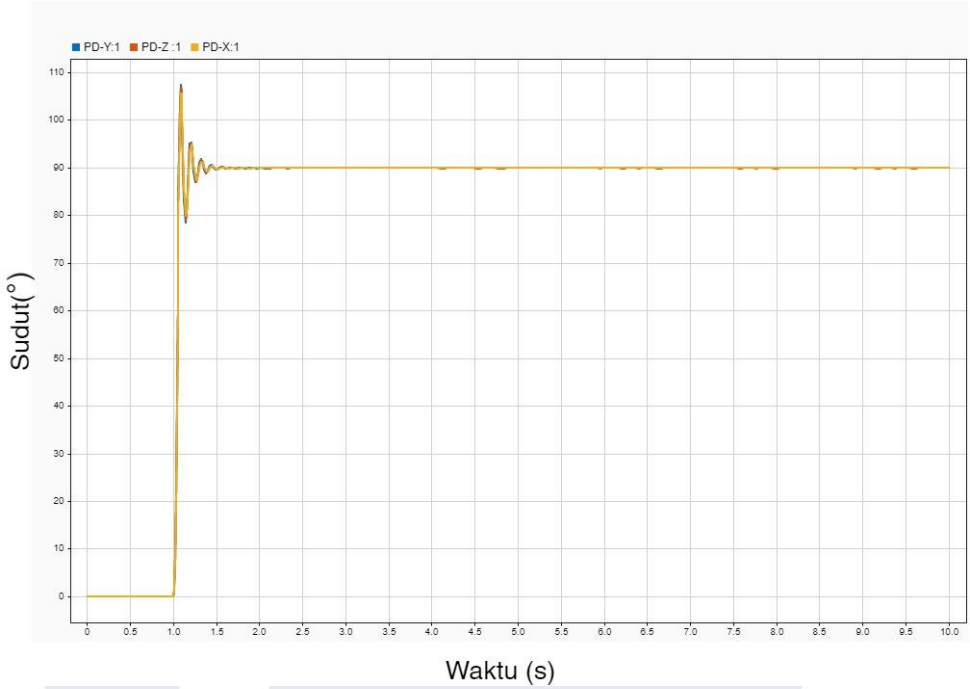


Gambar 3.16 *Step Response* PID Sumbu Y



Gambar 3.17 *Step Response* PID Sumbu Z

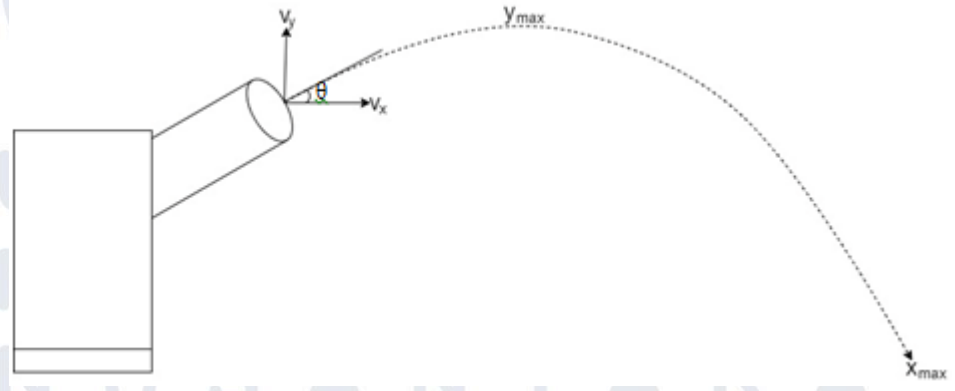
Di setiap *step response*, pengendali PD memiliki *overshoot* terendah, dan *settling time* yang cukup cepat. Pengendali dengan *overshoot* yang rendah dipilih untuk mengurangi *servo* bergerak secara dadakan ketika sistem baru dinyalakan.



Gambar 3.18 *Step Response* Tiap Sumbu

Ketika ketiga sistem kendali diberi *step function* dengan nilai 90^o seperti pada gambar 3.18, sistem kendali awalnya *overshoot* hingga 105^o – 108^o, lalu setelah sekitar 1.7 detik, sistem mulai stabil pada sudut 90^o. *Steady state error* sistem kendali tidak melebihi 1^o ketika sistem sudah stabil.

Secara *default*, *setpoint* setiap *servo* adalah 90^o. Artinya, PID akan menjaga posisi meriam agar selalu menghadap 90^o. Jika ORB mendeteksi target, *setpoint* berubah menjadi titik tengah target. Jika peluru dimasukan ke meriam, *setpoint* berubah berdasarkan hasil perhitungan parabola.



Gambar 3.19 Gerak Parabola

Gerakan peluru keluar dari meriam dapat dilihat pada gambar 3.19. Dalam gerakan parabola, seberapa jauh objek dapat terbang bergantung dari sudut elevasi, dan kecepatan awal. Kedua faktor ini akan mempengaruhi kecepatan horizontal, dan vertikal peluru. Dalam sistem ini, untuk membuat peluru terbang ke jarak yang diinginkan, sudut elevasi akan diubah dan kecepatan awal akan konstan pada jarak apapun. Perhitungan yang digunakan untuk mencari sudut elevasi adalah:

$$v_{0x} = v_0 cos\theta \quad (3.10)$$
$$v_{0y} = v_0 sin\theta \quad (3.11)$$
$$y = y_0 + v_{0y}t + \frac{1}{2}gt^2 \quad (3.12)$$
$$x = v_{0x}t \quad (3.13)$$

Di persamaan diatas, diketahui

v_0 = kecepatan inisial peluru (m/s)

v_{0x} = kecepatan inisial peluru pada sumbu horizontal (m/s)

v_{0y} = kecepatan inisial peluru pada sumbu vertikal (m/s)

y_0 = tinggi awal peluru saat dilontarkan (m)

y = tinggi akhir peluru setelah pelontaran (m)

x = jarak tempuh peluru pada sumbu horizontal (m)

t = waktu peluru terbang di udara (s)

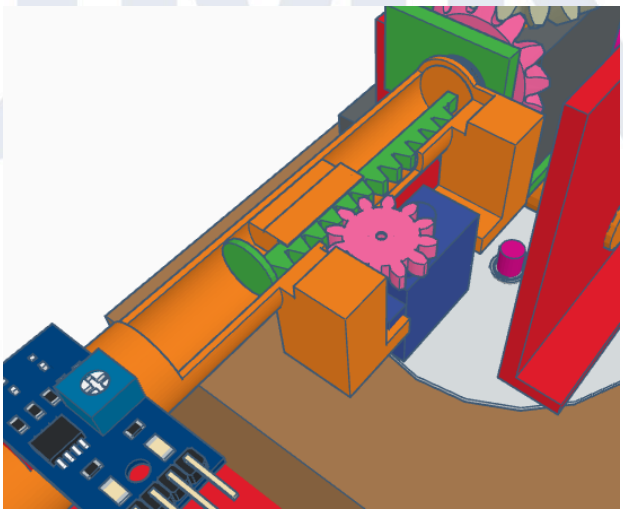
g = gravitasi ($9.8 m/s^2$)

Untuk memperoleh kecepatan inisial peluru, sudut elevasi diatur ke 0^o, lalu dilakukan penembakan beberapa kali untuk mendapatkan rata-rata jarak tempuh peluru pada sumbu horizontal. Karena sudut elevasi 0^o, kecepatan inisial peluru pada sumbu vertikal sama dengan nol, sehingga waktu peluru terbang di udara dapat diperoleh dengan perhitungan (3.12). Sudut elevasi 0^o juga artinya membuat kecepatan inisial pada sumbu horizontal sama dengan kecepatan inisial peluru. Dengan waktu peluru terbang di udara, dan rata-rata jarak tempuh peluru pada sumbu horizontal, kecepatan inisial peluru dapat ditemukan dengan perhitungan (3.13).

Dengan adanya kecepatan inisial peluru, perhitungan (3.10) – (3.13) dapat dilakukan dengan sudut berbeda. Jika hasil dari perhitungan (3.13) belum melewati jarak target, perhitungan diulangi dengan sudut ditambahkan 1^o. Jika hasil perhitungan (3.13) sudah melewati jarak target, nilai dari perhitungan (3.13) dibandingkan dengan hasil perhitungan menggunakan sudut sebelumnya untuk mencari jarak terdekat dengan jarak target. Sudut dengan hasil perhitungan (3.13) yang paling mendekati jarak target digunakan untuk menggerakan *servo* sumbu vertikal.

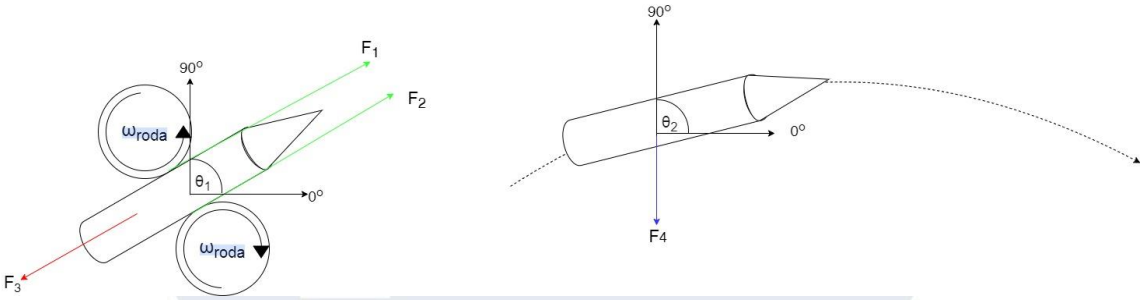
3.2.1.4 Hasil Implementasi Subsistem Pelontar

Subsistem diimplementasikan dengan menghubungkan L298N, TCRT5000, dan SG90 ke Arduino Mega. L298N dihubungkan ke dua motor yang terpasang di ujung laras meriam sebagai mekanisme pelontar. Sensor TCRT5000 dibaca secara *digital*, artinya *input* dari sensor berupa 0 atau 1. Meriam berwarna hitam sehingga sensor secara *default* akan memberi *input* "1". Ketika peluru masuk ke meriam, sensor akan memberi *input* "0". SG90 dipasang di meriam untuk mendorong peluru ke motor seperti pada gambar 3.20. Sebuah *gear* dipasang ke SG90, yang terhubung ke sebuah *rack*. *Rack* akan bergerak kedepan, dan mendorong peluru yang ada di ujung *rack*. SG90 hanya akan bergerak jika *input* dari TCRT5000 berupa "0".



Gambar 3.20 *Rack* dan SG90

Dua motor DC dipasang pada meriam sebagai pelontar peluru. Kedua motor berada pada kiri, dan kanan meriam dengan putaran yang saling berlawanan. Motor sebelah kiri akan berputar berlawanan arah jarum jam, dan motor sebelah kanan berputar searah arah jarum jam. Ujung kedua motor dipasang dengan sebuah ban RC yang terbuat dari karet. Gesekan yang dialami peluru akibat dua silinder yang berputar dengan arah yang berbeda akan membuat peluru meluncur keluar dari meriam.



Gambar 3.21 Gaya yang Dialami Peluru

$$F_t = F_1 + F_2 - F_3 - F_4 \quad (3.14)$$
$$F_1, F_2 = \mu_k ma$$
$$F_3 = mgsin\theta_1$$
$$F_4 = mgsin\theta_2$$

Dari persamaan gaya diatas, diketahui:

F_t = gaya total yang dialami peluru (N)

F_1, F_2 = gaya friksi antara ban dan peluru (N)

F_3 = gaya berat pada sumbu x (N)

F_4 = gaya gravitasi (N)

μ_k = koefisien gesek (0.7)

θ_1 = sudut ketika peluru dilontarkan (^{o})

θ_2 = sudut ketika peluru melayang di udara (^{o})

Pada pelontaran, terdapat tiga gaya utama yang dapat mempengaruhi total gaya yang dialami peluru. Pada saat pelontaran, gaya yang dialami peluru terdiri dari dua. Gaya gesek antara peluru dengan ban(panah hijau), dan gaya berat pada sumbu x(panah merah). Gaya gesek antara peluru dengan ban terpengaruhi oleh koefisien gesek. Untuk koefisien gesek

antara karet, dan plastik, diperkirakan nilainya sebesar 0.7. Ketika peluru keluar dari pelontar, kecepatan horizontal konstan berdasarkan kecepatan putar motor, sehingga hanya gerakan vertikal yang mengalami perubahan akibat gravitasi. Selama peluru belum mencapai tinggi maksimal, gaya gravitasi akan mengurangi total gaya. Ketika peluru melewati tinggi maksimal, gaya gravitasi akan menambahkan total gaya. Motor akan selalu berputar dengan kecepatan yang sama, sehingga untuk mencapai jarak yang diperlukan, hanya perlu diubah sudut elevasinya.

3.2.2 Hambatan Implementasi

Selama masa implementasi, terdapat berbagai masalah yang membuat sistem tidak dapat bekerja dengan yang direncanakan, atau membuat hasil percobaan tidak konsisten. Masalah yang ditemukan pada saat implementasi tidak hanya secara *hardware*, namun secara *software*. Penghubung laras meriam ke *gear* sumbu-x terlalu lemah sehingga laras meriam miring ke kanan. Ujung laras meriam juga terlalu berat, sehingga jika meriam bergerak terlalu cepat, *gear* sumbu-z bisa terlepas. Pada subsistem pengolah gambar digital, digunakan kamera DSLR sebagai *input frame*. DSLR dapat memberikan *live view* ke Raspberry Pi dengan menggunakan *library* gphoto2, atau dengan *capture card*. Namun, selama mencoba untuk mengimplementasikan subsistem, *live view* tidak dapat ditampilkan di Raspberry Pi.

3.2.3 Solusi yang Diterapkan

Untuk memperbaiki hubungan laras meriam ke *gear* sumbu-x agar tidak miring, diperlukan desain ulang dan *3D print* ulang penghubungnya. Mengetahui batas waktu, penulis memutuskan untuk menempelkan laras meriam langsung ke *servo* y. Artinya, *servo* y tidak dapat digunakan. Untuk masalah berat meriam, pengujian akan dilakukan dengan meriam didorong kebawah untuk memastikan meriam tidak lepas dari sumbu z.

Untuk hambatan *live view*, DSLR diimplementasikan dengan menggunakan *library* gphoto2. Sebuah program Python dibuat dengan fungsi untuk mengambil gambar setiap beberapa detik. Gambar tersebut kemudian

diubah namanya menjadi “InputImage.JPG” lalu dipindahkan dari *SD card* kamera ke *folder* di Raspberry Pi. Program ini akan dijalankan bersamaan dengan program pengolah gambar digital. Program pengolah gambar digital akan menggunakan ”InputImage.JPG” sebagai *frame* yang akan diproses, dan tiap beberapa detik “InputImage.JPG” akan diganti dengan gambar yang baru.